vertex^42

the excel nexus

" THE Guide to Excel in Business, Science, and Education "

Vertex42 Articles
- Excel Solver
- Monte Carlo Simulation
- > Histograms in Excel
- > Random Numbers
- Excel Web Queries
- Custom Excel

Microsoft Excel Solver add-in Exampl

- Jon Wittwer
Vertex42, LLC

The **Microsoft Excel solver add-in** is one of the features that makes creat engineering and financial models in a spreadsheet a powerful tool. To be cal "solver" doesn't do it justice, though, because it is really a powerful **optimiz** algorithm. The tool was developed by Frontline Systems, Inc. (Solver.com) offer a great deal of information on their website, including products that ex upon the free Excel solver add in. **This article provides a couple of exam how to use the Excel solver and call it using a VBA macro**.

Note: You do not need to download the add in. If you don't see it as one of the menu it Tools menu (in Excel), then you need to go to Tools > Add-Ins... and check the box nex Add In".

Excel Solver Examples

Example 1: "Finding a Local Minimum Using the Excel Sol

< Download Excel Solver Example 1 (.xls) >

Our first example is to going to be very basic, but it will introduce com terms used in optimization, such as ***objective function***, ***design vari*** and ***constraints***. Let's say we have the following equation, and we wa find the value of ***x*** that **minimizes *f*** subject to **-1 <= *x* <= 5**.

$$f = x^2 - x + 2$$

Our *objective function* is the value that we are going to minimize (*f*). T *design variables* are the variables that we are going to allow the Solve change (just *x* in this example). We have two *constraints*: **-1 <= *x*** an **5**

A convenient way of setting up this problem in Excel is to make a clea distinction between the objective, design variables, and constraints. A shot of the example problem is shown below, including the graph of th function so that you can see that **the answer should be somewhere between 0 and 2**. We need to choose a **starting value** for *x*, so let's *x* = 1 because that is the average number of times Excel crashes on m

Functions
Excel Training
Fun With Picture Links
Toolbar Buttons
Significant Figures

Excel Templates
> Amortization Table
> Asset Tracking
> Balloon Loan
> Excel Calendar
> Checkbook
> Class Schedule
> Expenses
> Inventory
> Invoice
> Mortgage
> Personal Budget
> Project Management
> Purchase Order
> Time Sheets
> Timelines
> Work Schedule
> Yearly Calendar
> ... Links ...

week.

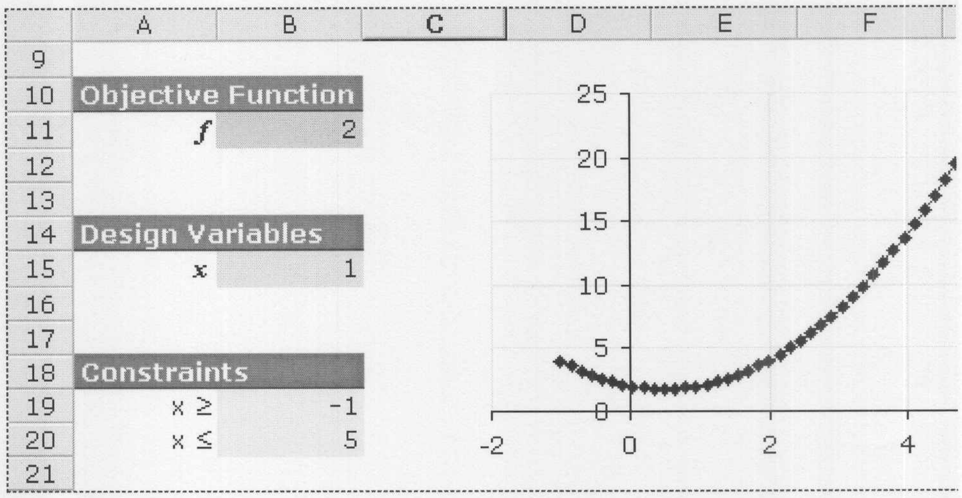


Figure 1: Screenshot of example problem 1.

Cell B11 (The Objective Function): **=B15^2-B15+2**

To use the Excel solver add in (Tools > Solver ...), we choose our obje function, cell B11, to be the "Target Cell" and choose the "Min" option Figure 2 below). Our only design variable is *x*, so the only cell we are change is B15. After adding the two constraints, we click on the Solve and we find our answer (*x*=0.5).

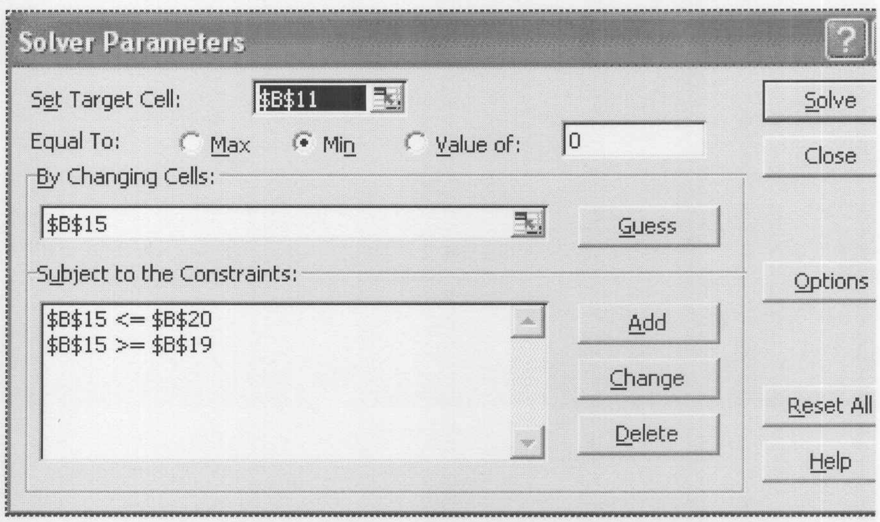


Figure 2: Screenshot of the solver add in dialog box for sample problem 1.

Example 2: "Solving a System of Non-Linear Equations"

< Download Excel Solver Example 2 (.xls) >

In this next practice problem, the solver is used to find values for the angles (θ_2 and θ_3) in the following system of equations.

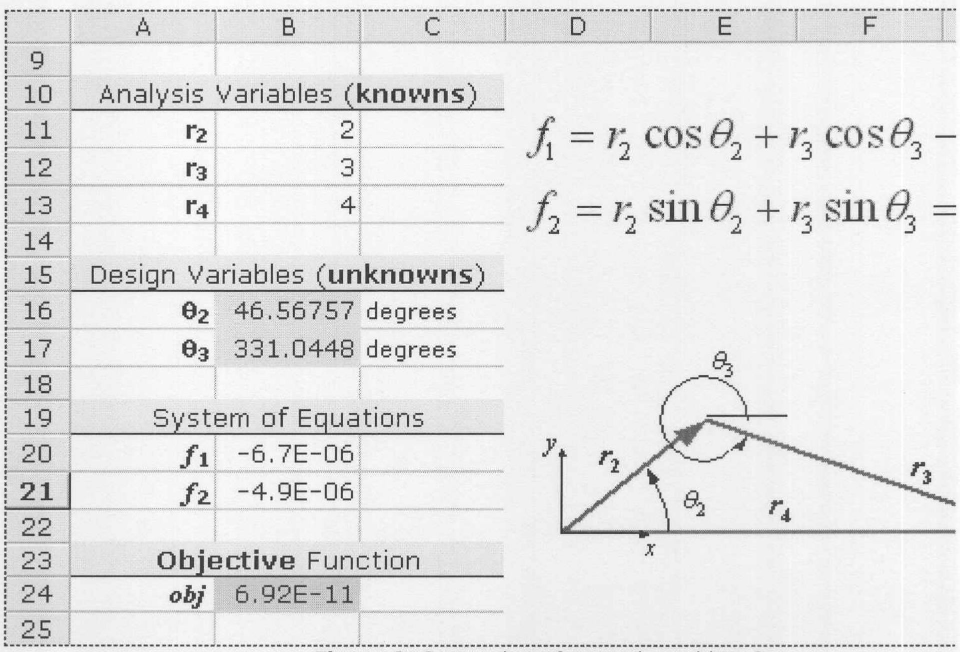
$$f_1 = r_2 \cos\theta_2 + r_3 \cos\theta_3 - r_4 = 0$$
$$f_2 = r_2 \sin\theta_2 + r_3 \sin\theta_3 = 0$$

Known: $r_2 = 2, r_3 = 3, r_4 = 4$

Unknown: θ_2, θ_3

Notice that these equations are in **implicit** form (equal to zero). To sc system, we will create an objective function that when **minimized, dr both equations to zero**. Minimizing the **sum of the squares** of eacl equation will accomplish this.

The layout for this problem is shown in the screenshot below. The kno variables are called ***analysis variables*** and will be treated as constar unknowns, θ_2 and θ_3, are the *design variables*. For this example proble don't have any constraints.



	A	B	C
9			
10	Analysis Variables (**knowns**)		
11	r_2	2	
12	r_3	3	
13	r_4	4	
14			
15	Design Variables (**unknowns**)		
16	θ_2	46.56757	degrees
17	θ_3	331.0448	degrees
18			
19	System of Equations		
20	f_1	-6.7E-06	
21	f_2	-4.9E-06	
22			
23	**Objective** Function		
24	***obj***	6.92E-11	
25			

Figure 3: Screenshot of example problem 2.

Is There Only ONE Solution?

The screenshot above shows one solution to the problem, but the solu depend upon the **starting values** that you have chosen for the unknc angles. For example, try using the starting values, θ_2 = -30 degrees, a 0 degrees. **You should get a different solution!** The figure below is example of a mechanism that can be described using these equations. second solution is represented by the dashed lines.

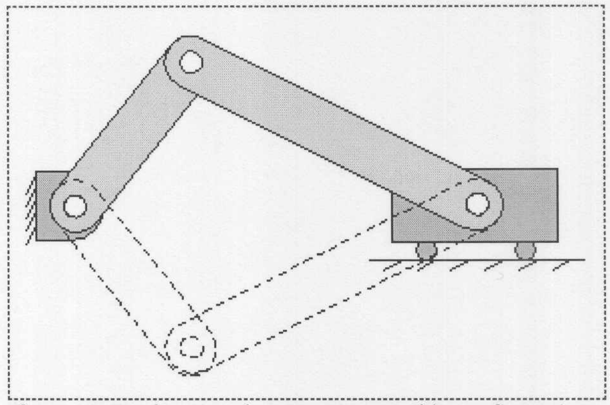
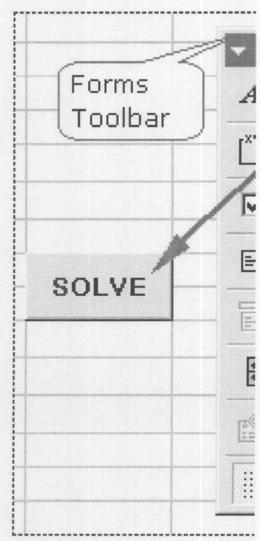


Figure 4: Mechanism showing two possible configurations.

This example has demonstrated a very important point having to do w Excel solver and optimization in general. **The solution may depend starting values**. For optimization problems, this means that the solut be only a **local** optimum.

Run the Solver Using a VBA Macro

If you need to solve the same system of equations or run an optimization ro number of times using the same model, it is convenient to create a macro tl be run by pressing a single button. An easy way to set this up is to first **rec the steps used to set up and run the solver**. Let's use the problem from 2 above.



To Record a Solver Macro:

- **Step 1:** Start the macro recorder (Tools > Macro > Record New Macro ...).
- **Step 2:** Open the solver dialog box (Tools > Solver ...).
- **Step 3:** Clear any existing solver settings (Press the **Reset All** button).
- **Step 4:** Choose the target cell, design variables, and constraints and press the Solve button. Then select OK to accept the results.
- **Step 5:** Stop the macro recorder (Tools > Macro > Stop Recording ...).
- **Step 6:** Add a button to the worksheet, using a button from the Forms toolbar. (If the Forms toolbar is not displayed, right-click on any toolbar and click on "Forms
- **Step 7:** Assign the macro you created to the button. (Right-click on t button and choose "Assign Macro ...")

Before the macro will work, a reference to the Solver VBA add-in functions r added.

Adding the Solver Reference:

- **Step 1:** Edit the macro you just created (Tools > Macro > Macros... (Alt+F8). This will open up Visual Basic. You can also press Alt+F11 to VBA.
- **Step 2:** Add the Solver reference in visual basic (Tools > References. make sure that SOLVER is checked).

The VBA code for the Solver macro that was recorded for Example 2 is below.

```
Sub SolverMacro()
' Example Solver VBA Macro
    SolverReset
    SolverOk SetCell:="$B$24", _
             MaxMinVal:=2, _
             ValueOf:="0", _
             ByChange:="$B$16:$B$17"
    SolverSolve userFinish:=True
End Sub
```

To keep the **Solver Results** dialog box from showing up, the **userFinish:=True** option has been added to the **SolverSolve** functio more help on using the Solver functions in VBA, search for "solver" us **VBA help** system.

REFERENCES:

"Creating Visual Basic Macros that Use Microsoft Excel Solver," From support.microsoft.com, July 24

"Solver Tutorial for Optimization Users," From www.Solver.com, July 24, 2004.

"Solver Uses Generalized Reduced Gradient Algorithm," *Microsoft Knowledge Base Article - 82890*, I support.microsoft.com, July 24, 2004.

CITE THIS ARTICLE AS:

Wittwer, J.W., "Excel Solver Examples" From *Vertex42.com*, July 24, 2004, http://vertex42.com/ExcelArticles/excel-solver-examples.html

